BAB III

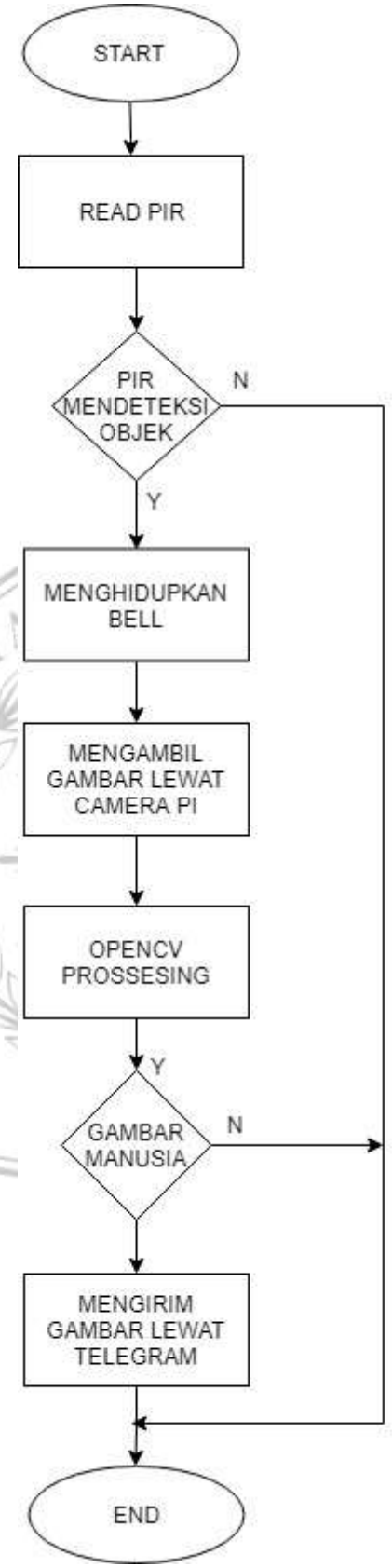
PERANCANGAN DAN PEMBUATAN ALAT

Pada bab ini akan di jelaskan sistematika pembuatan alat pemantau suatu rumah dan identifikasi objek manusia berbasis komputer mini yaitu *Raspberry pi*. Sistem ini dirancang di dalam rumah yang tidak berpenghuni atau penghuninya sedang sibuk bekerja yang di dalam rumah tersebut sangat lemah terhadap pengawasan atau mungkin penyusup. Sensor yang akan mendeteksi pertama kali adalah sensor PIR (*Passive Infra Red)*. Sistem akan mengambil inputan data dari sensor PIR dimana sensor ini menangkap objek dengan menganalisa suhu tubuh yang melewati sekitar rumah yang di pasangi oleh serangkaian alat ini.

Setelah sensor PIR (*Passive Infra Red)* dapat memberikan input data, maka *Raspberry pi* memberikan sinyal untuk membunyikan bell kemudian memberikan sinyal kembali ke *webcam* untuk mengambil gambar kemudian gambar ini akan di proses untuk mengidentifikasi bahwa di gambar yang di kirimkan itu adalah manusia atau bukan. Saat objek di identifikasi sebagai manusia maka *Raspberry pi* akan mengirimkan notifikasi berupa gambar ke *Telegram*.

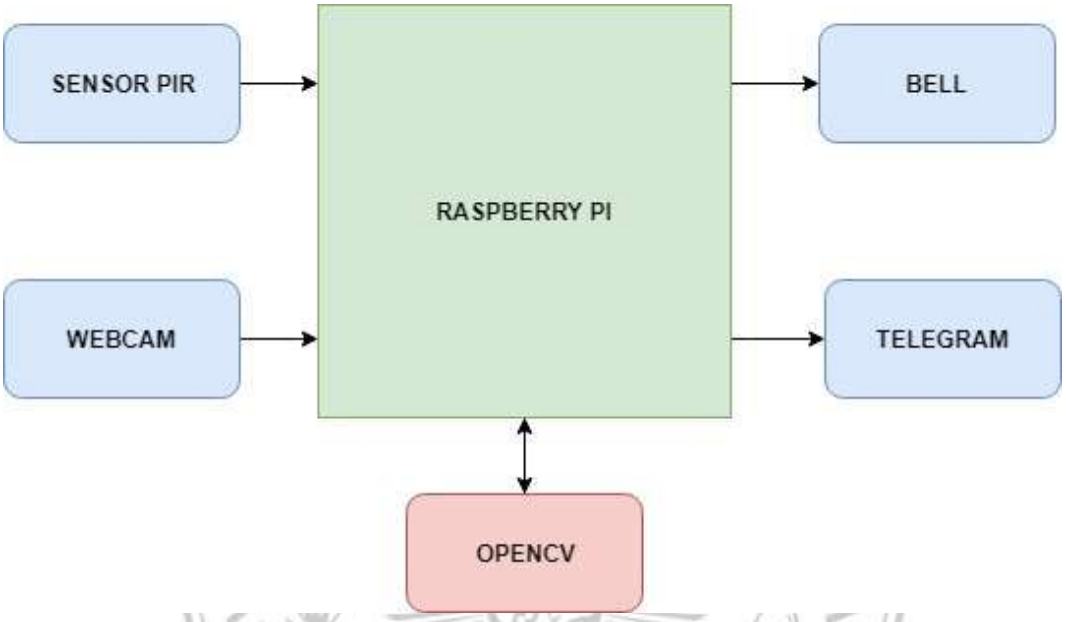
Pada proses pengidentifikasian gambar menggunakan *library Image Labelling* yang ada pada aplikasi *openCV* yaitu fungsi HOG *(Histogram Oriented of Gradient) descriptor*, maka suatu *image* dapat dideteksi objek tersebut atau manusia atau bukan. HOG *descriptor* sendiri merupakan suatu fitur dari gambar yang digunakan untuk menghitung vektor gradien pada area tertentu sehingga dihasilkan output berupa vektor yang nantinya diklasifikasi oleh *support vector machine.* [9]

Pada gambar 3.1 adalah *Flowchart* dimana *Flowchart* atau biasa juga disebut dengan diagram alir adalah sebuah jenis diagram yang didalamnya mempresentasikan algoritma secara detail serta prosedur sistem secara logika bisa dibilang diagram ini juga adalah tahapan instruksi tentang sebuah sistem secara berurutan.



Gambar 3. 1 Flowchart Sistem Otomasi Bel Rumah

Dari beberapa penjelasan dan juga *Flowchart* diatas maka bisa diperjelas lagi dengan adanya bagan pada gambar 3.2. Didalam bagan ini terdapat beberapa bagian yang akan dirancang, diantaranya yaitu terdapat *Raspberry pi, Raspberry pi, webcam,* sensor PIR, Bell , *OpenCV* dan yang terakhir adalah *Telegram.*



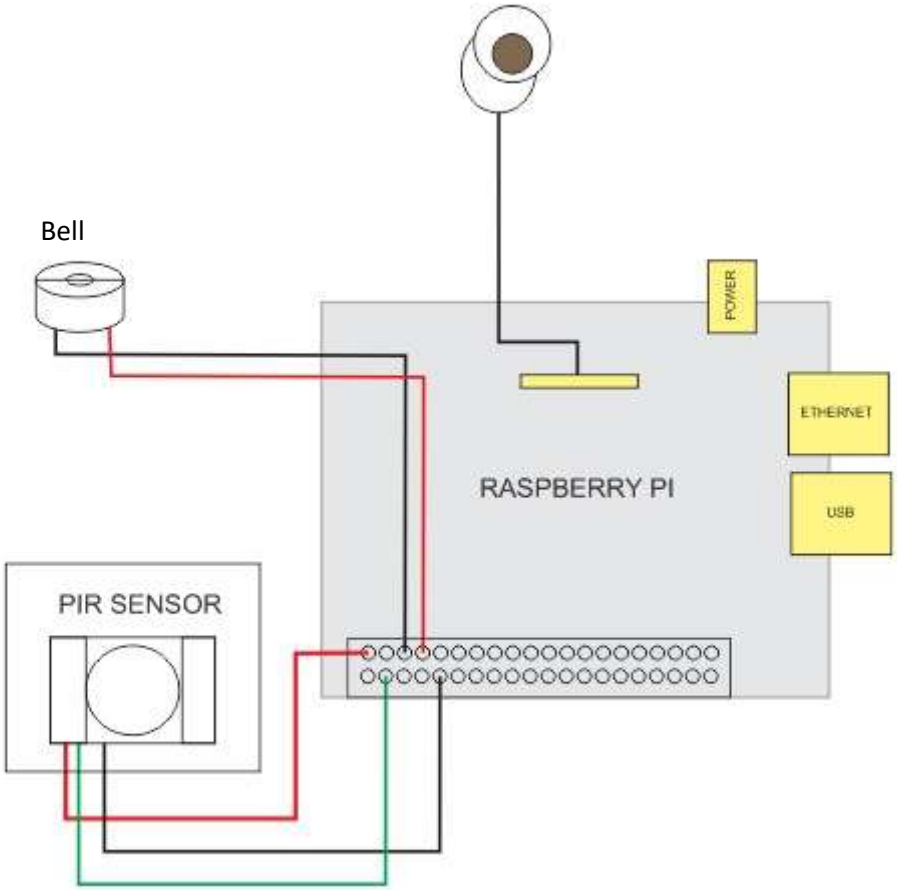
Gambar 3. 2 Blok Diagram Sistem

Adapun penjelasan yang ada pada gambar 3.2 dan gambar 3.1 diatas yang pertama yaitu Sensor PIR dengan spesifikasi HC-SR-04 yang diletakkan didekat pintu yang fungsinya untuk mendeteksi objek ketika ada benda (manusia atau bukan). Dan yang kedua adalah *camera Raspberry pi* , *camera* yang akan digunakan adalah *Raspberry pi camera* Rev 1.3 *camera* ini yang nantinya berfungsi sebagai pengambil gambar setelah sensor PIR mendeteksi adanya sebuah objek. Berikutnya adalah *OpenCV,* disini *OpenCV* berfungsi sebagai alat untuk bisa mendeteksi wajah. Dan yang ke empat adalah *Raspberry pi, Raspberry pi 3* tipe B yang akan digunakan untuk alat ini. *Raspberry pi* disini berfungsi sebagai penghubung antara *camera pi* dan juga *OpenCV.* Selanjutnya adalah bell dimana bell yang digunakan adalah bell yang berjenis *Piezoelectric,* karenakan bell jenis ini memiliki berbagai macam kelebihan dibanding bell tipe yang lainnya seperti kalau dilihat dari harganya relatif lebih murah, lebih ringan dan juga dalam hal penggabungannya kedalam rangkaian elektronika lainnya relatif lebih mudah.

Setelah dilihat dari alur diatas bisa diperjelas lagi dengan gambar 3.2 perancangan alat dibawah ini.

3.1 Perancangan Alat

Perancangan alat atau biasa disebut perancangan perangkat keras *(Hardware)* adalah beberapa komponen perangkat keras yang berada didalam sebuah komputer yang secara fisik dapat untuk dirasakan, diraba dan juga dapat untuk dilihat. [4]

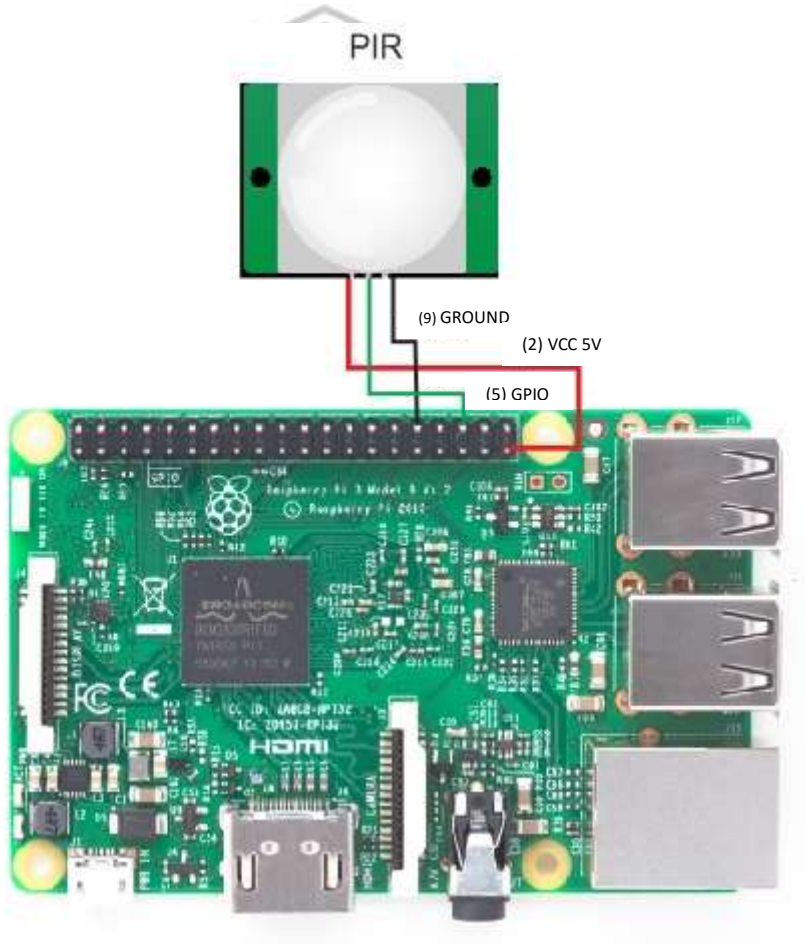


Gambar 3. 3 Perancangan Alat

Didalam masalah ini perancangan alat dibagi menjadi beberapa bagian, diantaranya yaitu perancangan dari sensor jarak atau PIR, perancangan dari modul *webcam*, dan yang terakhir adalah perancangan dari bell. Dari gambar 3.3 tentang perancangan alat maka dapat dijabarkan lagi seperti dibawah ini.

3.1.1 Perancangan sensor PIR HC-SR04

Perancangan sensor PIR pada kali ini digunakan untuk mengetahui keberadaan objek di depan pintu. Sensor secara otomatis membaca data ketika mendeteksi objek dengan jarak 5 meter. Sensor ini memiliki 4 pin yaitu *VCC, trigger, echo, grounding.* Kemudian pada pin *echo* akan mengeluarkan sinyal *output* berupa pulsa dan nantinya akan dibaca oleh *Raspberry pi* . Pin *echo* pada PIR akan dihubungkan pada pin 3 di *Raspberry pi,* pin ini adalah pin GPIO kemudian pin VCC akan dihubungkan dengan pin 2 dan pin ini adalah pin catu daya dengan besar tegangan 5 volt.

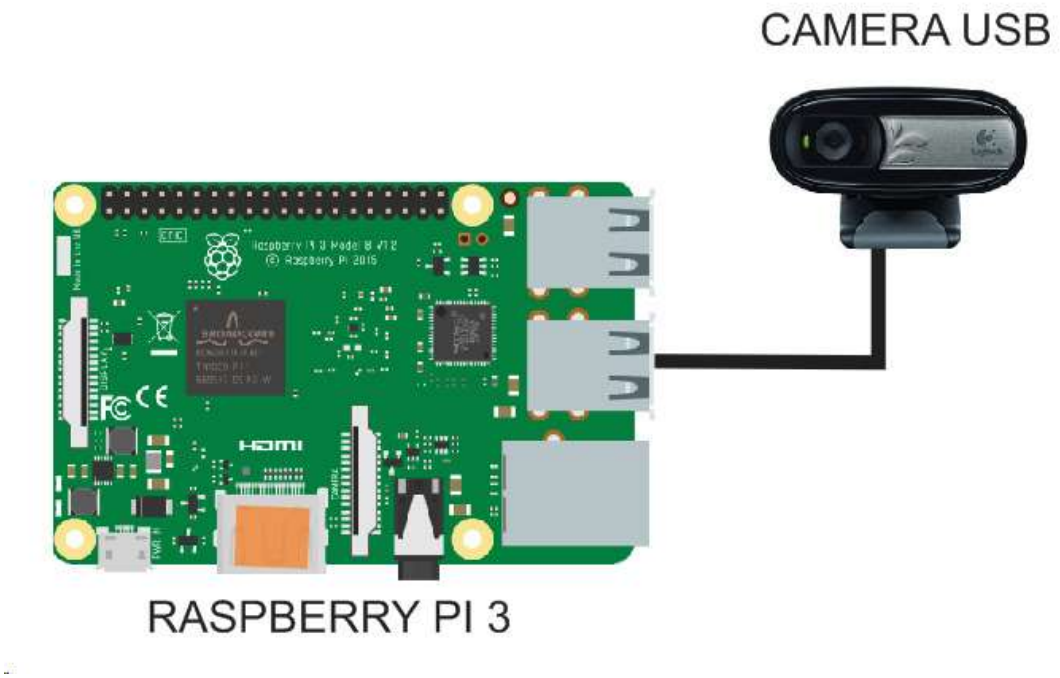


Gambar 3. 4 Perancangan Sensor PIR

3.1.2 Perancangan Rangkaian Modul Webcam

Perancangan modul *Webcam* dapat terkaitkan secara langsung dengan sistem pada *Raspberry pi* dengan cara mengatur *port* di *webcam* yang akan

digunakan atau di *setting. Driver* untuk *webcam* harus di instalasi terlebih dahulu sebelum digunakan, karena aplikasi presensi akan dikoneksikan ke *deriver webcam* utuk memfungsikannya sebagai *input* media. Jika sudah di instalasi maka langkah selanjutnya tinggal menghubungkan *webcam* ke *port* USB di *Raspberry pi,* maka *webcam* siap digunakan menjadi media *input* sistem dan juga tentunya kualitas pengambilan sebuah gambar yang dapat di *setting* untuk semaksimal mungkin dengan cara menetapkan spesifikasi yang semaksimal mungkin pada *webcam.* Untuk lebih lengkapnya bisa dilihat dari gambar 3.5.

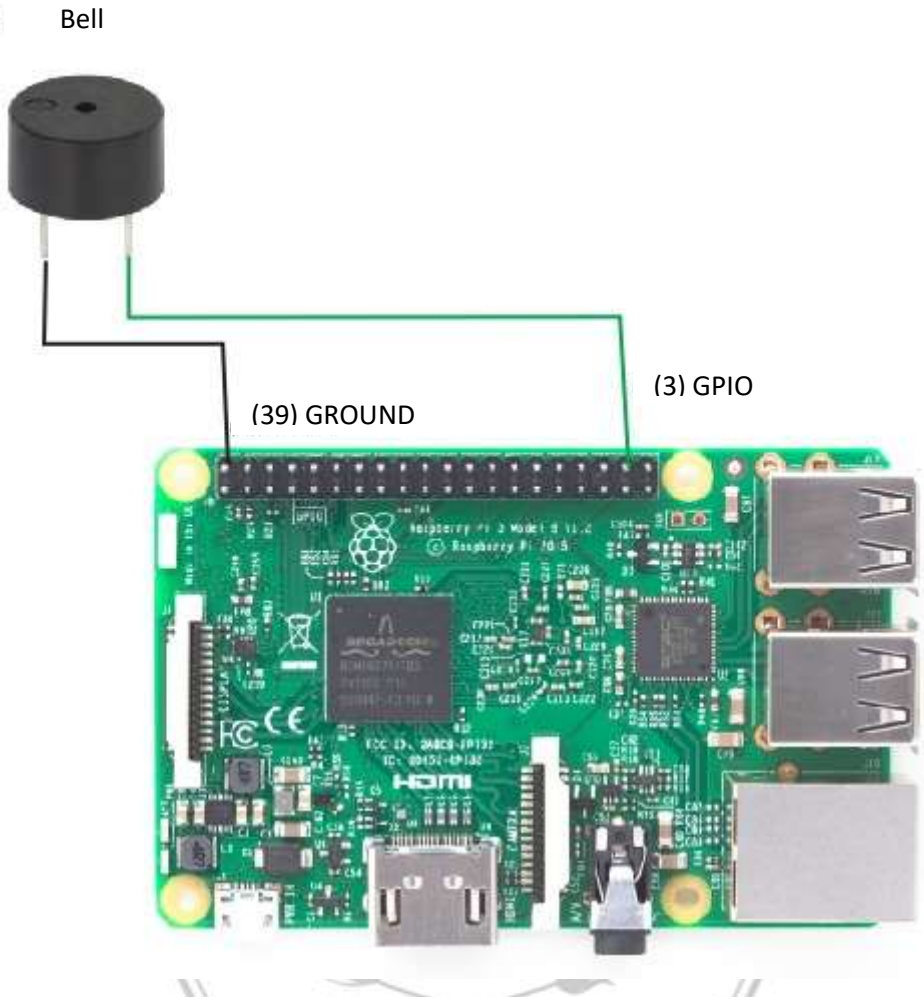


Gambar 3. 5 Perancangan Rangkaian Modul Webcam

3.1.3 Perancangan Bell

Bell berfungsi sebagai indikator bunyi pada sistem ini. Bell ini adalah sebuah alat yang terdapat kumparan elektromagnetik didalamnya dimana fungsi dari kumparan ini adalah untuk mengubah sinyal listrik menjadi sinyal getaran suara. Bell adalah sebuah perangkat audio yang sering digunakan pada sebuah rangkaian anti maling, bel rumah, peringatan waktu mundur pada mobil dan pada perangkat bahaya lainnya. Bell memiliki 2 kaki, pin 1 dihubungkan ke port GPIO

pada *Raspberry pi*. Kemudian pin 2 di hubungkan ke GND. Untuk lebih lengkap dan juga jelasnya bisa dilihat dari gambar 3.6 dibawah ini.



Gambar 3. 6 Perancangan Bell

3.2 Perancangan Perangkat Lunak (Software)

Software atau biasa disebut dengan perangkat lunak komputer adalah sebuah perangkat yang didalamnya berisi tentang serangkaian instruksi, program, prosedur, pengendali pendukung dan beberapa aktivitas pengolahan perintah pada sebuah sistem di komputer. Jadi, dapat disimpulkan bahwa *software* adalah sebuah komponen yang abstrak dari susunan sebuah sistem pada komputer. [4]

Dalam perancangan *Software* atau perangkat lunak ini terdapat beberapa perancangan yang dibutuhkan agar alat dapat bekerja dengan optimal, yang pertama yaitu Instalasi *Raspbian OS,* yang kedua yaitu Instalasi *Raspbian OS* dan yang

terakhir yaitu perancangan sistem notifikasi. Maka dapat dijabarkan sebagai berikut dibawah ini.

3.2.1 Instalasi Raspbian OS

Raspbian OS adalah sebuah sistem operasi yang dikhususkan untuk perangkat yang berbasis *IoT*. Operasi sistem ini berjalan pada operasi sistem berbasis *Linux Distro Debian.* Pada sistem operasi ini akan dipasang pada sebuah penyimpanan *micro SD* dan pada pemasangan ini dilakukan menggunakan sebuah aplikasi *Win32disklmager*. Awalnya penulis mengunduh sebuah *master* sistem operasi *Raspbian OS* di laman *https://www.raspberrypi.org/downloads/* kemudian setelah mengunduh lewat aplikasi yang ada di *Win32disklmager* ke penyimpanan berupa *micro SD* yang akan digunakan untuk *Raspberry pi*. Mengingat spesifikasi perangkat keras yang diusung pada komputer mini *(Raspberry pi)* ini terbatas sehingga *Raspbian OS* yang penulis pilih sebagai penunjang dari *miniPC* agar *miniPC* bisa berjalan dengan optimal. Operasi sistem ini juga *support* terhadap aplikasi *OpenCV* yang akan berfungsi sebagai pendeteksi wajah.

3.2.2 Instalasi OpenCV

Perancangan sistem pendeteksi objek menggunakan aplikasi *OpenCV*. Dimana aplikasi ini adalah aplikasi yang digunakan untuk mengolah data citra digital. Gambar yang diambil oleh *camera*, kemudian akan di proses oleh *script* untuk menentukan objek ini adalah manusia atau bukan. Didalam *OpenCV* disediakan banyak algoritma visi pada komputer dasar dan *OpenCV* juga menyediakan modul untuk mendetekdi objek. Dengan menggunakan *library* pada *OpenCV* yaitu pada fungsi HOG *(Hystogram Oriented of Gradien) descriptor*, maka suatu *image* akan dapat di deteksi bahwa itu manusia atau bukan. HOG *descriptor* merupakan sebuah fitur gambar yang akan digunakan untuk menghitung sebuah vektor gradien pada sebuah area tertentu sehingga menghasilkan *output* berupa vektor yang akan diklasifikasikan oleh *support vector machine.* [4]

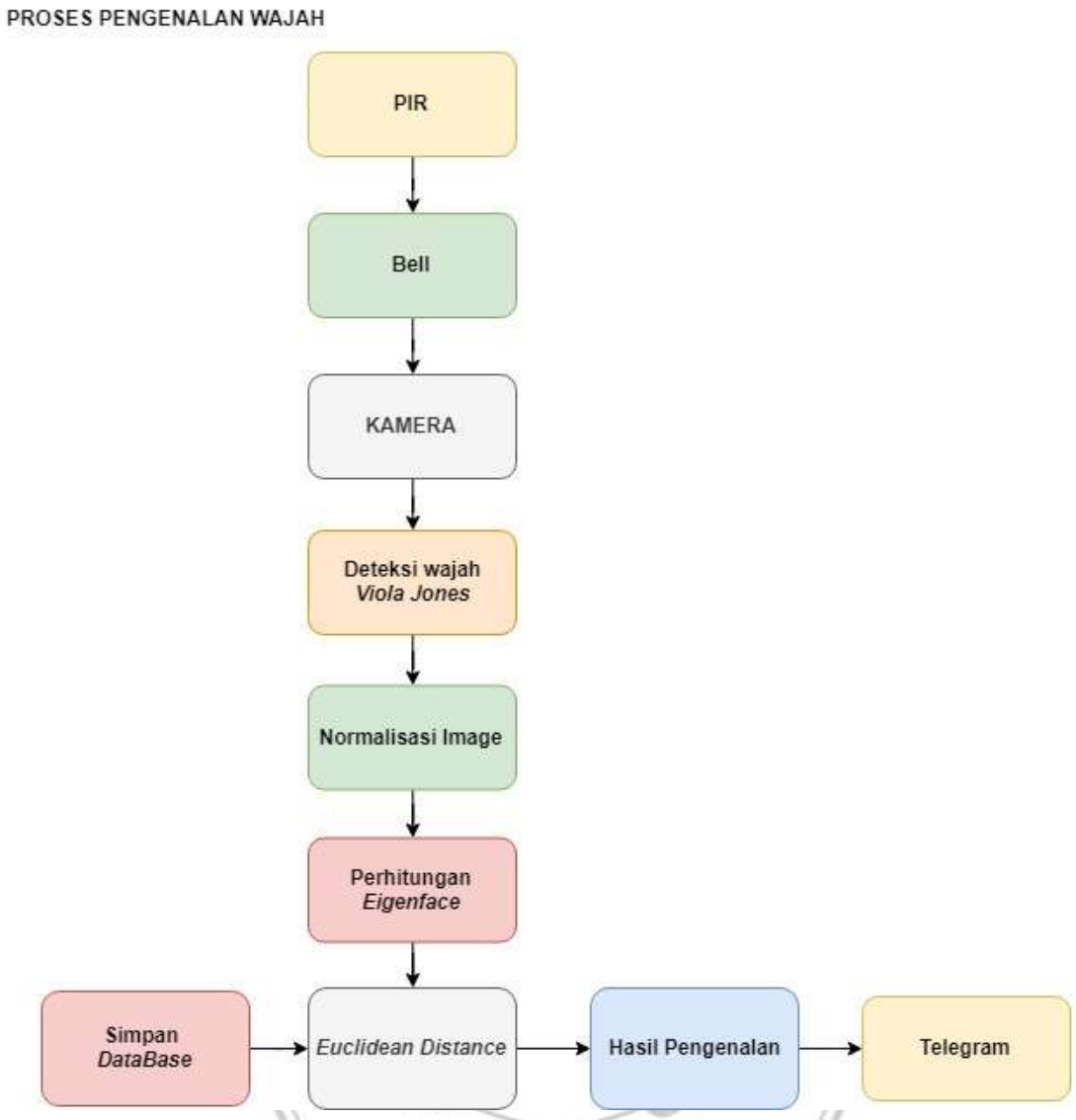
Pada tahap proses instalasi aplikasi *OpenCV* dibutuhkan koneksi ke *internet* karena beberapa paket pendukung aplikasi *OpenCV* harus di instal melalui *internet.* Ada bebrapa paket yang perlu di instal terlebih dahulu antara lain :

- build-essential
- git
- cmake
- pkg-config
- libjpeg8-dev
- libtiff4-dev
- libjasper-dev
- libpng12-dev
- libavcodec-dev
- libavformat-dev
- libswscale-dev
- libv4l-dev
- libgtk2.0-dev
- libatlas-base-dev
- gfortran
- pythin2.7-dev

Proses instalasi dari paket-paket tersebut melalui terminal degan cara mengetikkan *apt-get install,* kemudian mengunduh *master*

3.2.3 Perancangan Sistem Pendeteksi Wajah

Didalam perancangan sistem pengenalan wajah ini diharapkan mampu untuk mengenali beberapa wajah yang sudah tersimpan didalam *database* dengan akurat dan juga cepat. Berikut adalah diagram blok tentang bagaimana sistem pengenalan wajah ini bekerja.



Gambar 3. 7 Diagram Blok Pengenalan Wajah

Pada gambar blok diagram 3.7 dapat dijelaskan pada sistem ini adalah blok dari proses pengenalan dari hasil karakteristik dan keluaran berupa notifikasi telegram dan bunyi bell. Didalam blok proses untuk pengenalan wajah sendiri dan juga output berupa *notifikasi telegram* dan juga bell teridiri dari kamera dan peragkat lunak yang akan memproses *Input* citra menjadi sebuah karakteristik citra yang tadi disimpan didalam *database* sehingga nantinya akan memberikan sebuah *Output* hasil pengenalan berupa pengenalan yang kemudian akan menghasilkan suara bell dan juga *notifikasi* pada *telegram.*

Gambar dari wajah baru atau *image* dari sebuah wajah baru atau biasa disebut dengan *test face* akan dicoba untuk dikenali, penerapan cara pada proses awal atau proses yang pertama yaitu perhitungan dari *eigenface* untuk mendapatkan nilai *eigenface* dari sebuah *image* tersebut.

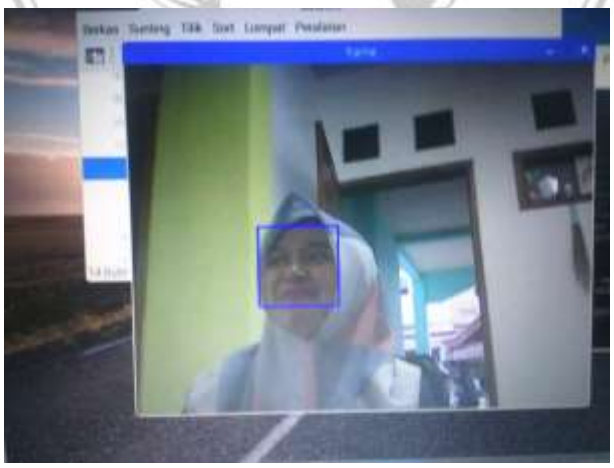
1. Sensor PIR aktiv ketika sensor mendeteksi adanya sebuah gerakan lalu kemudian sensor menghidupkan bell.
2. Setelah sensor PIR aktiv dan menghidupkan bell maka *Webcam* akan aktiv agar dapat menampilkan gambar yang telah ditangkap oleh *webcam* kedalam aplikasi. Deteksi wajah sendiri menggunakan metode *Viola Jones* akan dilakukan secara langsung *(real time)* menggunakan *webcam.* Dengan kombinasi *Cascade of Cassifier* dan juga *Adaptive Boosting* akan mempercepat proses pendeteksian objek yaitu berupa wajah. Jika wajah telah terdeteksi akan dilakukan penggambaran berupa garis persegi pada wajah yang telah terdeteksi.
3. Setelah dilakukan deteksi wajah, selanjutnya yaitu dilakukan pengkapan citra wajah atau biasa disebut dengan *Image Capturing* dapat juga dilakukan secara langsung *(real time)* menggunakan *webcam.*
4. Selanjutnya yang dilakukan yaitu sebuah proses awal yang akan meliputi, normalisasi dari ukuran citra, RGB dirubah ke *grayscale*, selanjutnya untuk memperbaiki kualitas citra *input* agar lebih memudahkan proses pengenalan wajah tanpa menghilangkan informasinya yang utama yaitu menggunakan *histogram equalization*, setelah itu dilakukan *resize* yang fungsinya adalah untuk membuang bebrapa bagian daerah selain wajah sehingga hanya pada bagian wajah saja yang akan di proses dan yang akan di normalisasi pencahayaan ketika pengambilan *Input* citra.
5. Dilakukan sebuah proses perhitungan untuk mengutip pada bagian terpenting dengan menggunakan metode *Eigenface* sehingga didapatkanlah *Eigenvector* dan *Eigenvalue* dari *Image* tersebut.
6. Kemudian dilakukan proses penyimpanan kedalam data wajah pada setiap citra wajah yang digunakan dalam sebuah proses perhitungan kedalam bentuk *.xml, semakin sering dan kompleks maka proses dari pengenalan wajah maka akan semakin baik.

7. Dari beberapa data yang telah disimpan nantinya akan digunakan sebagai nilai pembanding dari proses perhitungan jarak untuk pengenalan wajah.
8. Setelah itu dilakukanlah proses perhitungsn untuk mengtip bebrapa bagian terpenting dengan menggunakan metode *Eigenface* sehingga nanti akan didapatkan *Eigenvector* dan juga *Eigenvalue* dari *Image* tersebut.
9. Untuk proses pengenalan wajah dengan menghitung jarak diantara fitur wajah yang berada didalam data dan juga fitur dari wajah yang baru. Jarak yang akan diidentifikasi adalah jarak yang didapat yang paling kecil.

3.2.4 Deteksi Wajah menggunakan metode Viola Jones

Deteksi wajah menggunakan metode *Viola Jones* dapat dilakukan secara langsung atau *real time* menggunakan *webcam.* Dengan menggunakan kombinasi *Cascade of Classifer* dan juga *Adaptive Boosting* dapat mempercepat proses pendeteksian objek berupa wajah manusia. Jika objek wajah telah terdeteksi, maka dilakukan penggambaran sebuah garis persegi pada wajah yang telah terdeteksi. *Image Capturing* dapat dilakukan secara *real time.*

```
faces = detector.detectMultiScale(gray, 1.3, 5)
    for (x,y,w,h) in faces:
            cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
```



Gambar 3. 8 Gambar Deteksi Wajah Menggunakan Metode Viola Jones

Tahapan Intergral Image

Di dalam tahapan *Integral Image* adalah cara untuk menghitung persegi panjang secara cepat dengan menggunakan rumus :

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', yi)$$

. . . . . (3.1)

Dari rumus diatas maka didapat skript yang ada dibawah ini :

```
def my_intergral_image(img):
    intergral_img = np.zeros(img.shape)
    for x in range(img.shape[1]):
        for y in range(img.shape[0]):
            for i in range(x + 1):
                for j in range(y + 1):
                    intergral_img[y, x] += img[i, j]
    zero_padded_intergral_image = np.zeros((img.shape[0] + 1, img.shape[1] + 1))
    zero_padded_intergral_image[1:img.shape[0] + 1, 1:img.shape[1] + 1] = intergral_img
    return zero_padded_intergral_image
```

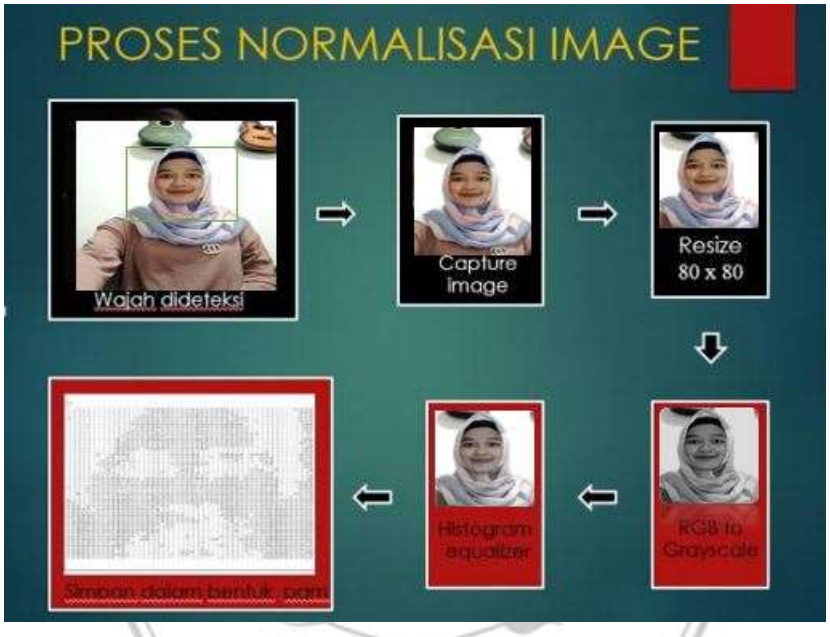
3.2.5 Normalisasi Gambar (Image)

Normalisasi gambar dilakukan awal dengan meliputi, normalisasi dari ukuran gambar atau citra, merubah dari RGB menjadi *grayscale,* memperbaiki *input* kualitas citra agar lebih mempermudah proses pengenalan wajah tanpa harus

menghilangkan informasi utamanya dengan menggunakan *Histogram Equalization.* Kemudian di *Resize* untuk membuang beberapa bagian di daerah selain wajah sehingga hanya di bagian objek wajah saja yang akan di proses dan juga di normalisasi oleh pencahayaan pada saat proses pengambilan *input* citra. Dan selanjutnya yang dilakukan yaitu menyimpan data *input* wajah yang telah diambil kedalam bentuk *.JPG.

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

cv2.imwrite ("dataSet/User."+Id +'.'+ str(sampleNum) + ".jpg", gray [y:y+h,x:x+w])



Gambar 3. 9 Proses Normaliasi Image

3.2.6 Proses EigenFace

Berikutnya adalah proses *EigenFace* dilakukan untuk mengutip beberapa bagian yang penting karena dari proses ini nantinya akan didapatkan *Eigenvector* dan juga *Eigenvalue* dari gambar yang sudah didapat tadi. Untuk setiap citra wajah yang telah digunakan dilakukan proses penyimpanan data wajah dalam proses *EigenFace* didalam bentuk *.yml, semakin sering dan kompleks proses pengenalan wajah maka hasil akan semakin baik.

faces,Ids = getImagesAndLabels('dataSet')

recognizer.train(faces, np.array(Ids))

recognizer.save('trainner/trainner.yml')

Gambar 3. 10 Data Training *.yml.

3.2.7 Proses Includian Distance

Didalam proses *Includian Distance* adalah proses dimana dari data wajah dari hasil proses training akan dibandingkan atau dicocokkan dengan data dari data wajah yang baru untuk dicari nilai yang paling terdekat. Data wajah yang baru akan dilakukan pencarian *Eigenvector* dan akan dicari nilai yang paling terdekat yang sekiranya mendekati nilai yang sudah tersimpan didalam *database* sehingga nanti akan didapatkan hasil dari pengenalan wajah tersebut.

```
recognizer = createEigenFaceRecognizer()
recognizer.load('trainner/trainner.yml')
cascadePath = "viola_jones.xml"
faceCascade = cv2.CascadeClassifier(cascadePath);
for(x,y,w,h) in faces:
        cv2.rectangle(im,(x,y),(x+w,y+h),(225,0,0),2)
        Id, conf = recognizer.predict(gray[y:y+h,x:x+w])
        if(conf<100):
                Id = names[ids.index(str(Id))]
```

3.2.8 Perancangan Sistem Notifikasi

Pada perancangan sistem notifikasi dibutuhkan aplikasi *Yowshap* . Aplikasi ini di instal di sistem operasi *Raspbian OS. Yowshap* akan mengirimkan informasi dari *Raspberry pi* ke pengguna *Telegram* melalui program *(BOT)* yang di tugaskan untuk mengambil data dan dikirimkan secara otomatis tanpa ada pihak ketiga yang memerintahkan. Listing program sebagai berikut.

Import telepot

Bot=telepot.Bot('724016941:AAE3KywKAxbjfFv_ELOA1A62xAb52XYuvWc')

chat_id1 = 784444292

bot.sendMessage(chat_id, “text”)